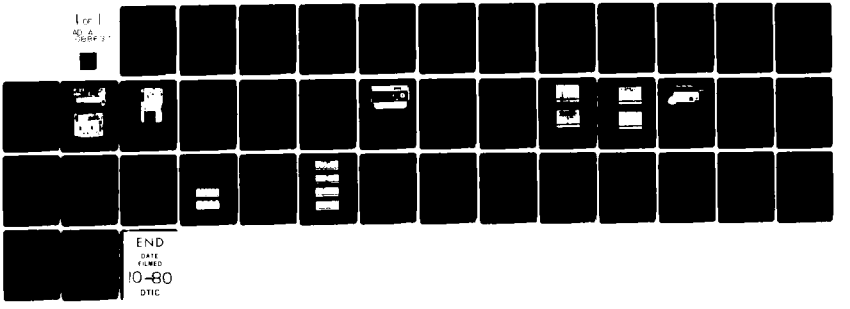
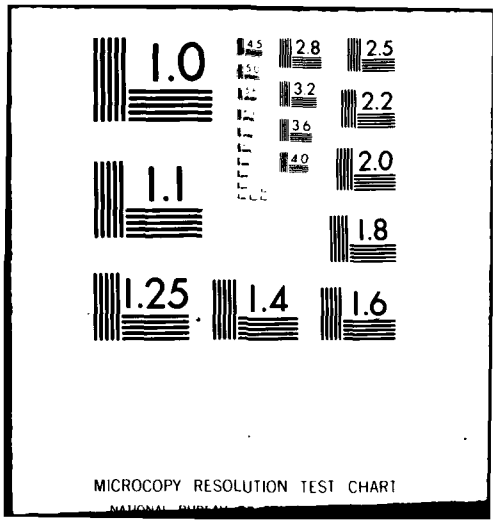
AD-A088 637 ARMY ELECTRONICS RESEARCH AND DEVELOPMENT COMMAND FO--ETC F/G 9/2
MICROCOMPUTER INTERFACING TECHNIQUES.(U)
JUL 80 J T CERVINI
UNCLASSIFIED DELEW-TR-80-1 NL



END
DATE FILMED
10-80
DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
MICROCOPY RESOLUTION TEST CHART

RESEARCH AND DEVELOPMENT TECHNICAL REPORT

DELEW-TR-80-1

MICROCOMPUTER INTERFACING TECHNIQUES

John T. Cervini
ELECTRONIC WARFARE LABORATORY

July 1980

ERADCOM

US ARMY ELECTRONICS RESEARCH & DEVELOPMENT COMMAND
FORT MONMOUTH, NEW JERSEY 07703

NOTICES

Disclaimers

The citation of trade names and names of manufacturers in this report is not to be construed as official Government indorsement or approval of commercial products or services referenced herein.

Disposition

Destroy this report when it is no longer needed. Do not return it to the originator.

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: DELEW-TR-80-1	2. GOVT ACCESSION NO.: AD-A088 637
3. RECIPIENT'S CATALOG NUMBER:	
4. TITLE (and Subtitle): MICROCOMPUTER INTERFACING TECHNIQUES	5. TYPE OF REPORT & PERIOD COVERED: Final 2-01-79 thru 2-01-80
	6. PERFORMING ORG. REPORT NUMBER:
7. AUTHOR(s): JOHN T. CERVINI	8. CONTRACT OR GRANT NUMBER(s): 11 Jul 80
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Commanding General, US Army Electronics Research & Development Command, ATTN: DELEW-E, Fort Monmouth, NJ 07703	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: IT162715-A042-04-02
11. CONTROLLING OFFICE NAME AND ADDRESS:	12. REPORT DATE: July 1980
	13. NUMBER OF PAGES: 37
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Final rept. 1 Feb 79-1 Feb 80	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT (of this Report)

Distribution is unlimited; approved for public release.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

MC6800-Microprocessor, Microcomputer, Interface, Data Acquisition, assembly language.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A number of parallel interfacing methods for microcomputer systems are discussed which use the Motorola 6800 microprocessor as a central processing unit. A minimum processing system is defined, and the individual electronic components are defined and explained. The report also illustrates the requirements of the Peripheral Interface Adapter integrated circuit in effecting the Input/Output of data. The techniques and instructions necessary to interface signal generators, printers, and data acquisition systems to a microcomputer are also described.

CONTENTS

	Page
INTRODUCTION	
DISCUSSION	
Microcomputing Systems	
PIA Interface Requirments	
Signal Generator Interface	13
Printer Interface	16
Data Acquisition Systems	26
CONCLUSIONS	36
BIBLIOGRAPHY	37

FIGURES

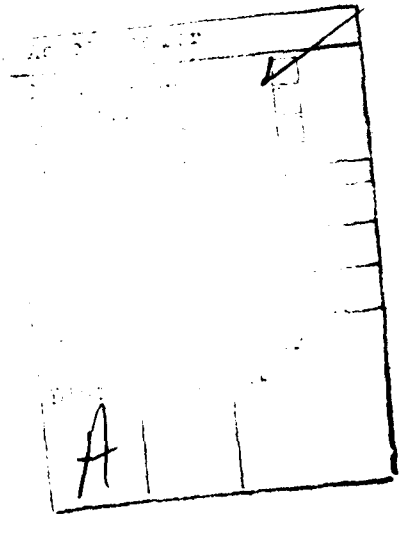
1.	Minimum Microcomputer Configuration.	2
2.	MC 6800 Microprocessor Block Diagram.	3
3.	MCM 6810A Random Access Memory (RAM).	4
4.	MCM 6830A Read Only Memory (ROM).	5
5.	MC 6820 Peripheral Interface Adapter (PIA) Block Diagram.	6
6.	MCM 6810A RAM bus interface.	7
7.	MCM 6830A ROM bus interface.	7
8.	MC 6820 PIA bus interface.	8
9.	Motorola MEK 6800 D1 Evaluation Module.	9
10.	Motorola MEK 6800 D2 Evaluation Module.	9
11.	Motorola MEK 6800 D2 Display Board.	10
12.	RS 232C format (30 characters/sec.).	10
13.	Peripheral Interface Adapter (PIA) functions.	12
14.	WAVETEK Model 3001 Signal Generator.	14
15.	Signal generator block diagram.	14
16.	Rear panel connector and pin ID.	15
17.	Signal generator outputs.	17,18
18.	DPP-7 Digital Panel Printer.	19
19.	DPP-7 block diagram.	20
20.	Handshaking with the printer on the "B" side.	21
21.	APP-20 block diagram.	23
22.	APP-20 timing - printing one line.	24
23.	Pulse output on "B" side configuration.	25
24.	CB2 outputs in pulse mode.	25
25.	CB2 inverted pulses at various intervals.	27
26.	Successive Approximation A/D Converter.	28
27.	Dual - slope integrating waveform.	29
28.	SHM-4 Sample and Hold Circuit.	30
29.	N-Channel multiplexer circuit.	31

Page

30. ADC-149 S/A A/D Converter. 32
31. Handshaking with peripheral on "B" side. 34

TABLES

1. APP-20 I/O Pin Connection. 21
2. APP-20 I/O Interface Connections. 22
3. MUX Channel Addressing. 35



MICROCOMPUTER INTERFACING TECHNIQUES

INTRODUCTION

The state-of-the art in electronics has advanced to the point where a number of "smart" processors exist which can be applied to an infinite number of applications. The microprocessor unit (MPU) is rapidly replacing both digital and analog circuitry in the industrial control environment. At the same time, the MPU is beginning to challenge the once exclusive domain of the minicomputer manufacturers, who themselves, are challenging the main-frame producers.

Military designers and planners should become aware of the power and flexibility of the microprocessor and its ability to lower the cost of designing electronic equipment. These assets can be realized with the additional benefit of increased capability. The current processing power of the MPU, in general, is quite wide. However, each individual microprocessor operates within its own limited realm, based on the size of its address bus and data word. This report will concentrate on the Motorola 6800 MPU which has an 8-bit data word and a 16-bit address bus that can access 2^{16} or 65,535 locations.

Microprocessors by themselves are not usually self-contained units, although "single-chip" MPU's now exist. Strictly speaking, the microprocessor is an incomplete processing module, consisting of an arithmetic and logic unit (ALU), an internal instruction set, internal registers, and input/output (I/O) buffers. In order to be useful, the MPU needs a number of peripheral circuits to allow it to communicate with the outside world. A minimum configuration consists of an MPU, read/write memory, an interface circuit, and an external clock system (Fig. 1). This total package would be defined as a microcomputer.

This report deals specifically with microcomputer configurations based on the Motorola 6800 MPU and its family of circuits. Many of the techniques explored here are general enough to be applied to other MPU families, such as the Rockwell 6502 microprocessor. A tutorial will not be given on the understanding of MPU technology; a basic knowledge of microprocessors and assembly language programming will be assumed. Not all interface applications will be treated in this report. It will concentrate on parallel interface techniques, leaving serial interface developments, such as modems, to other investigations.

DISCUSSION

Microcomputing Systems

The minimum systems microcomputer configuration, as shown in Fig. 1, is physically represented by individual silicon chips assembled onto Dual Inline Packages (DIP's). The design engineer who wishes to fabricate a custom microcomputing system for a specific application must design the

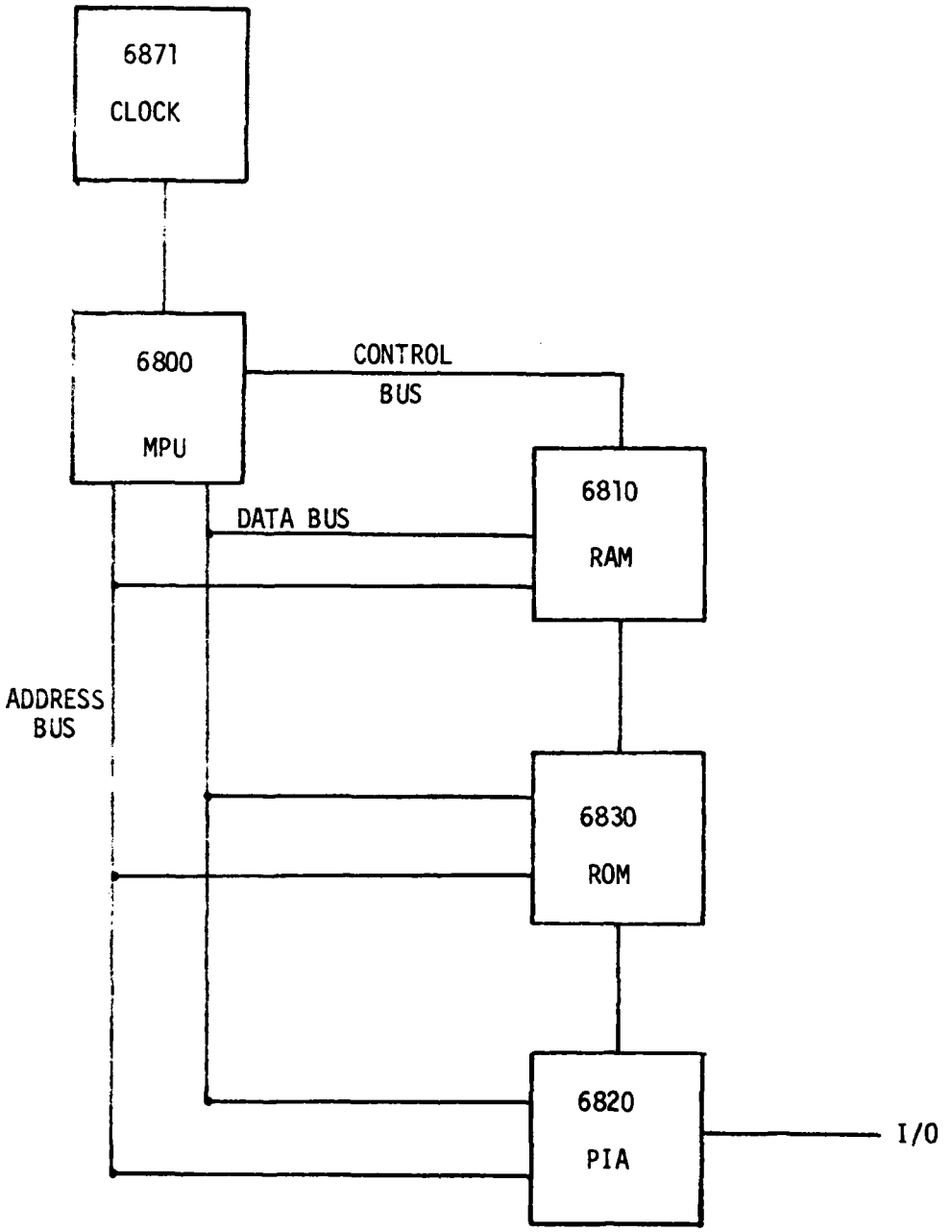


Figure 1. Minimum Microcomputer Configuration.

proper connections for the address, data, and control buses. Next he must put together the various hardware components, program the necessary software, and finally test and debug the entire system. To accomplish this feat, the design engineer must have a working knowledge of the integrated circuits (IC's) needed in the design.

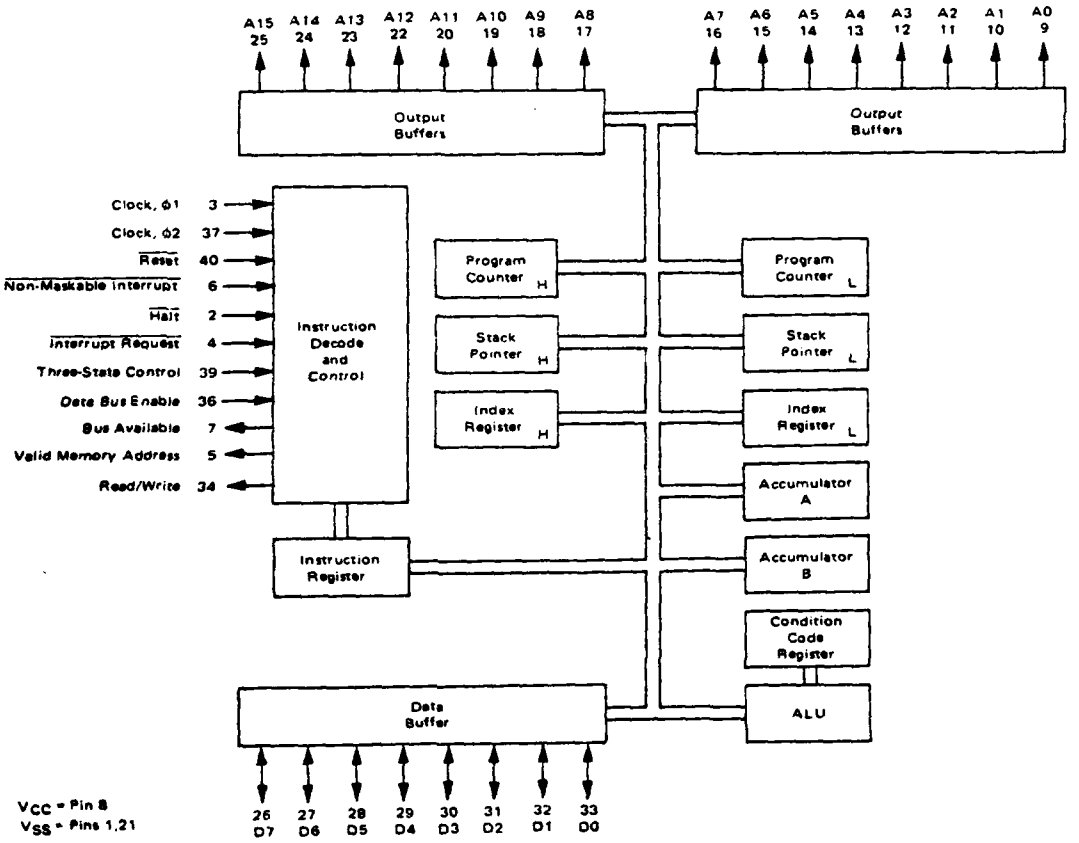


Figure 2. MC 6800 Microprocessor Block Diagram

Figure 2 is an expanded block diagram of the MC 6800 microprocessor.[1] The processor is a bidirectional, bus-oriented, 8-bit (data word) parallel machine with 16 bits of address. The MC 6800 has two 8-bit accumulators, A and B, which are used to hold operands and results from the Arithmetic Logic Unit (ALU). The 16-bit index register, X, stores 16 bits of memory address for the index mode of memory addressing. The stack pointer is a two byte (8-bits/byte) register that contains the address of the next available location in an external push-down/pop-up stack. The stack is normally a random access read/write memory that may have any location (address) that is convenient. The program counter is a 16-bit register that contains the current program address. The condition code register is a flag register which contains 6 bits to indicate the results of an ALU operation. Processor control lines include Reset, which automatically restarts the processor; Interrupt Request; and Non-Maskable Interrupt to

1. M6800 Microcomputer System Design Data, Motorola Inc., 1975

monitor peripheral status. Finally, there is a Three-State Control, Data Bus Enable, and a Halt Control Line, which can be used for Direct Memory Access (DMA) or multi-processing.

Read/write memory for the MPU can be provided by one or more Random Access Memory (RAM) chips. For dedicated control applications, where only a small amount of memory is needed to store intermediate results, the MCM 6810 device is easily used. This chip is a 128x8-bit totally static RAM, which was designed to operate in bus-organized systems (Fig. 3). The static property of this device allows it to function without being periodically "refreshed" by the MPU; i.e., it retains data in a location as long as power is applied to the circuit.

Read Only Memory (ROM) for programmed instruction is provided by the MCM 6830 1024x8-bit IC (Fig. 4). This device is a "masked" chip, which means that the memory locations are pre-programmed by the manufacturer with a specific set of instructions. This is done when the ROM is to be used as a limited function operating system. More will be said about this later. The design engineer will normally use an ultraviolet (UV), electrically programmable, read-only memory (EPROM). In any case, a ROM cannot be written into, but can only be read from. Also, the MCM 6830 is a static byte oriented device designed to operate with the MC 6800 MPU.

The MC 6820 Peripheral Interface Adapter (PIA) provides the universal means of interfacing peripheral equipment to the MC 6800 microprocessor. A block diagram is presented in Fig. 5. This device is capable of interfacing the MPU to peripherals through two 8-bit bidirectional peripheral data buses (PA0-PA7 and PB0-PB7) and four control lines (CA1, CA2, CB1, CB2). The functional configuration of the PIA is programmed by the MPU during system initialization. Each of the peripheral data lines can be programmed to act as an input or output; each of the four control/interrupt lines can be programmed for one of several control modes.

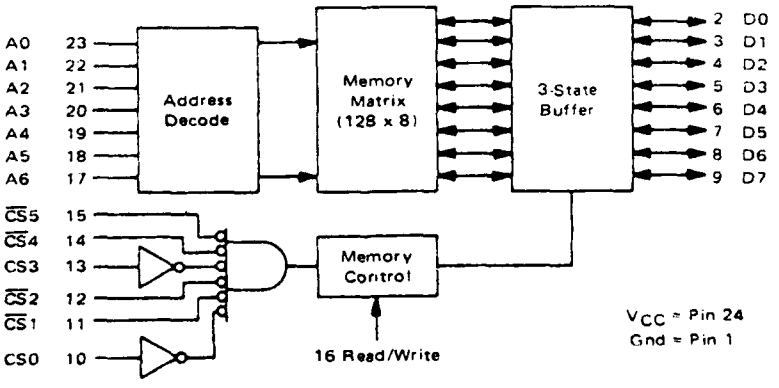


Figure 3. MCM 6810A Random Access Memory (RAM).

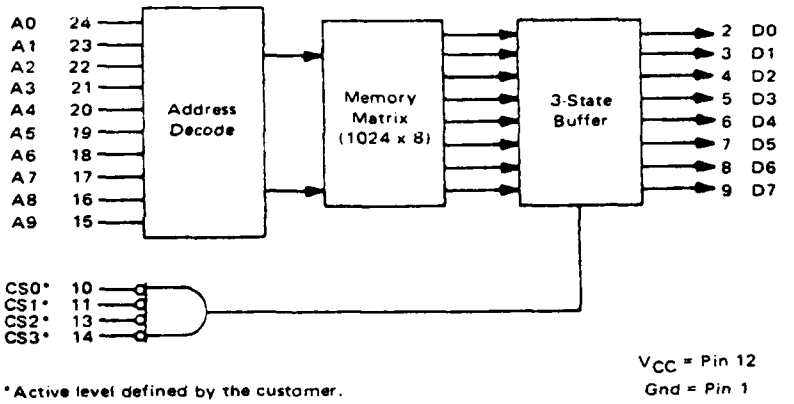


Figure 4. MCM 6830A Read Only Memory (ROM).

The interconnection of these components to form a microcomputer system is relatively straightforwrd with the MC 6800 device. Since all components operate at the same transistor-transistor logic (TTL) levels and with the same drive capability, the data, address, and control lines can be interconnected without adding external TTL buffers. Figure 6 shows the 6810 RAM bus interface to the MPU and the rest of the components of the system. The RAM has six Chip Select (CS) inputs, four active-low and two active-high, which interface directly to the address bus. Four of the CS's are used to decode the system address lines. In small- and medium-sized systems, this address decoding will be sufficient to distinguish between all integrated circuits in the system without using any additional address decoding packages.

The ROM bus interface (Fig. 7) is similar to that of the RAM, and all outputs may be connected directly to the data bus without drivers. It has 10 address lines and four CS's, three of which are used to provide the address decoding that selects a particular location within the ROM.

The PIA system interface is shown in Fig. 8. It has three CS lines and two Register Select (RS) lines, RS0 and RS1, connected to the address bus. Two Interrupt Request (IRQ) lines, IRQA and IRQB, are used to signal the MPU when an interrupt has occurred. The data bus lines, Chip Select, Read/Write, and Enable have the same static and dynamic characteristics as the other peripherals in the MC 6800 system. The Reset line is used to initialize the PIA. The RS lines serve the same purpose in the PIA as the address lines do in memory; they address the control and status registers, thereby making the PIA look like memory to the microprocessor.

Although the interfacing of components is in general straightforward, it can be somewhat time-consuming and tedious to design a system from scratch. Fortunately, there are products on the market that can ease this burden because they have hardware already designed and debugged. This simplifies writing the desired program to accomplish a particular task. Some difficulties will still arise since some type of development system will be needed to debug the program in order to insure correctness. We can take things a step further by using a microcomputer evaluation module, which

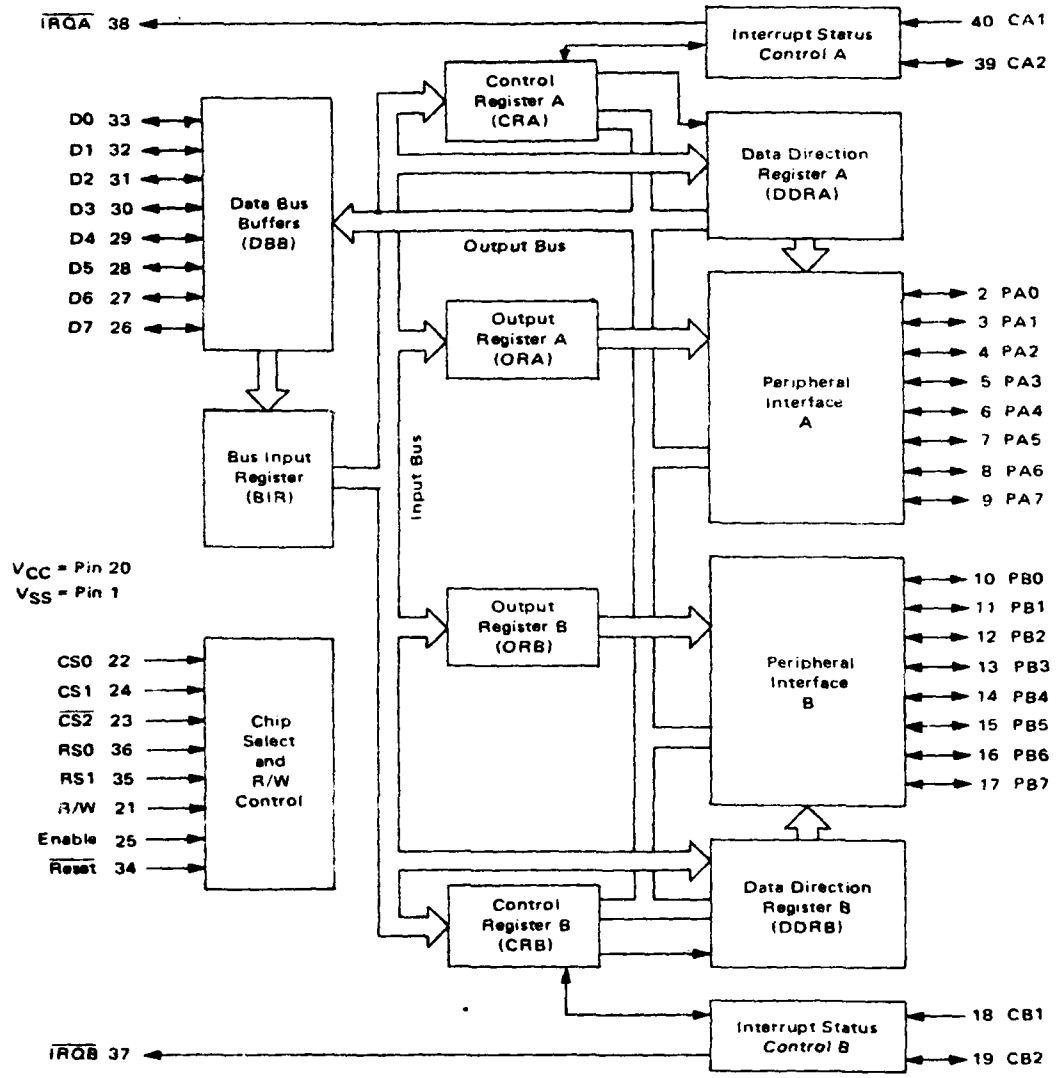


Figure 5. MC 6820 Peripheral Interface Adapter (PIA) Block Diagram.

essentially consists of designed, fabricated, and debugged hardware, and an associated monitor integrated circuit chip which debugs the software. The monitor chip is usually in the form of a pre-programmed ROM circuit, which is part of the hardware design. The monitor ROM functions as a very low-level operating system (OS) since it normally maintains overall control of the entire system.

A microcomputer evaluation module represents one of the simplest true computer systems. It can be used for introductory teaching purposes or can serve as a breadboard prototype for more sophisticated systems. The main function for our use is in dedicated control applications.

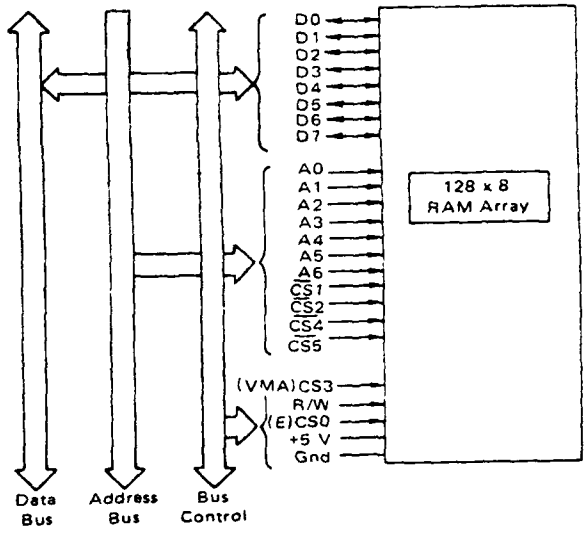


Figure 6. MCM 6810A RAM bus interface.

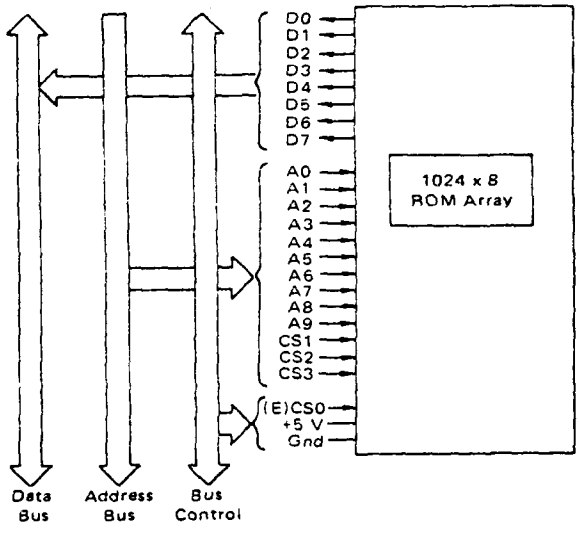


Figure 7. MCM 6830A ROM bus interface.

Examples of microcomputer evaluation modules are Motorola's MEK 6800 D1 module (Fig. 9) and MEK 6800 D2 (Figs. 10 and 11). The MEK 6800 D1 module is a single-board microcomputer system designed to interface with a teletype or a computer terminal. The printed circuit (PC) board contains a 6800 MPU, two PIA's, one ACIA, six RAM's, and one ROM, plus a number of additional IC's, resistors, and capacitors necessary for operation. The ROM is the MIKBUG (trademark of Motorola Inc.) monitor which provides an asynchronous communications program, a loader program, and a diagnostic program for use with the 6800 MPU. The PC board can be configured to use either a teletype

(TTY) interface or an RS-232 interface for communicating with different terminals at various band rates. The keyboard of the RS-232 or TTY device provides the means of communicating with the MIKBUG monitor. Software routines in the ROM echo each character as it is typed on the keyboard. The monitor's Punch and Read commands allow the transfer of data to and from the PC board and cassette or paper tape. The RS-232C standard format for 30 characters per second (300 baud) is shown in Fig. 12.

FIGURE 4 – MC6820 PIA BUS INTERFACE

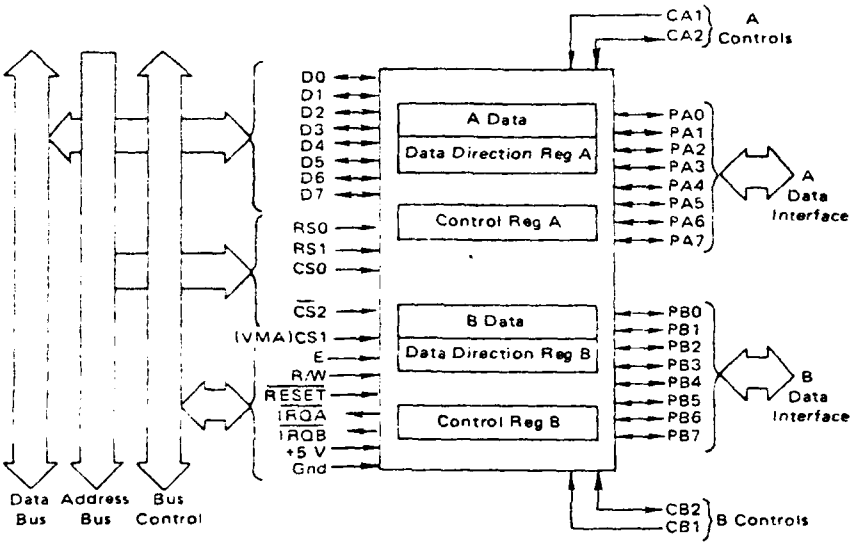


Figure 8. MC 6820 PIA bus interface.

An evaluation module provides the means necessary to compose, debug, and execute machine language software using the existing hardware. The program to be executed is initially stored in RAM memory. It is either loaded from tape or entered through a keyboard, using the Memory (M) change command of the monitor. Then it is executed. If the program doesn't work, it can be debugged, using the Register (R) command. The R command will display the contents of the accumulators, X-register, condition code register stack pointer, and the program counter so that the programmer can check for proper operation of the software.

The JBUG (trademark of Motorola Inc.) monitor of the MEK 6800 D2 evaluation module was primarily designed to input data through a keyboard and display it on a series of seven-segment digits (Fig. 11). There is also a provision for transferring data to and from a cassette recorder through a special "Kansas City" interface. This interface is used mainly for bulk transfer of longer programs to and from RAM memory. JBUG contains additional commands, which greatly enhance the operator's ability to debug software. The break-point insertion and removal command, V, allows break points to be set at selected places in the program. Part of

Figure 9. Motorola MEK 6800 D1 Evaluation Module.

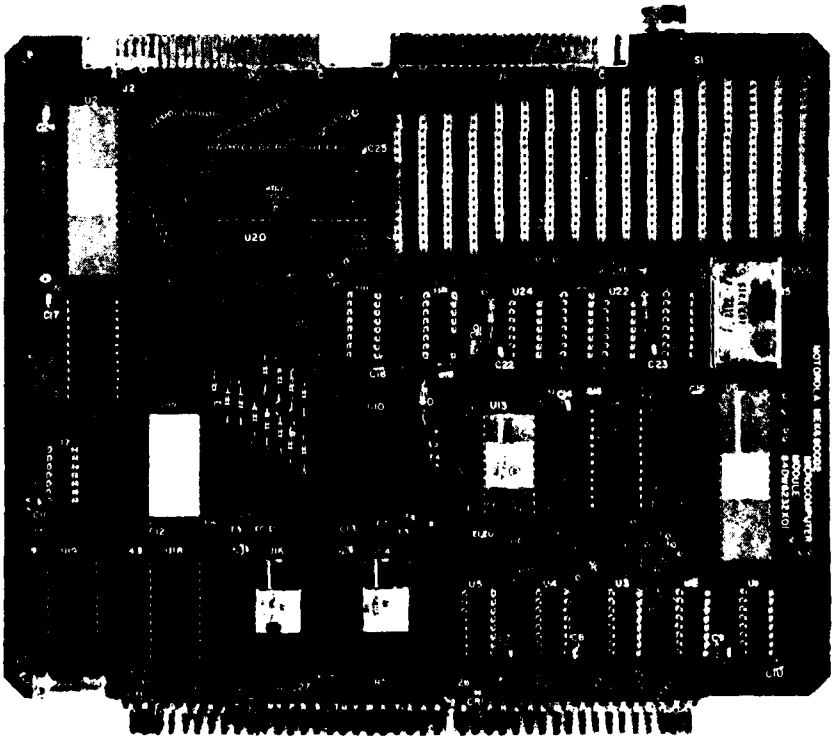


Figure 10. Motorola MEK 6800 D2 Evaluation Module.

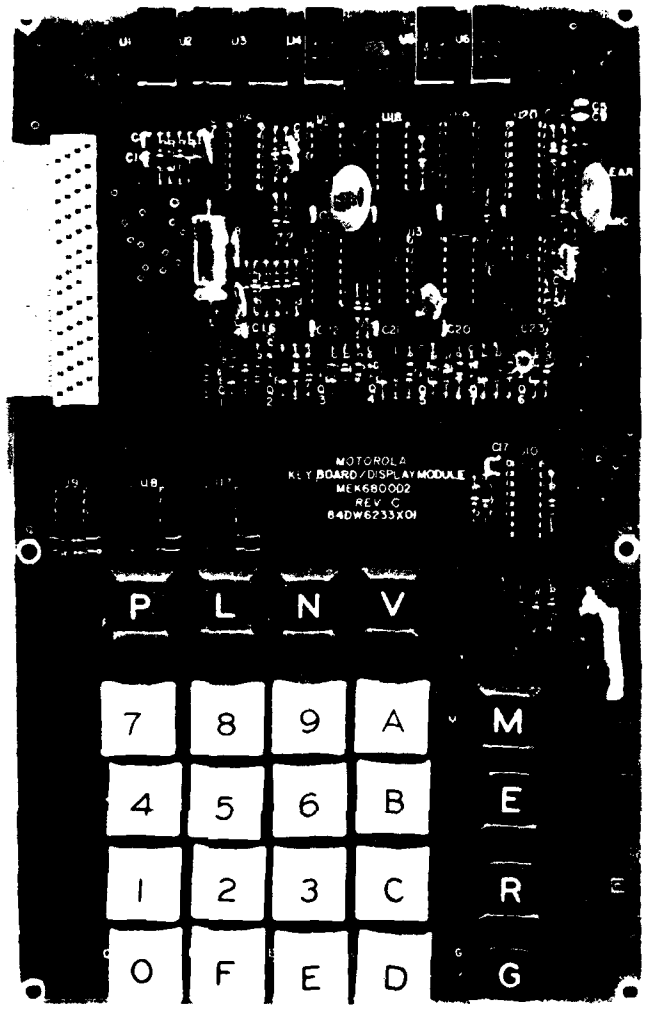


Figure 11. Motorola MEK 6800 D2 Display Board.

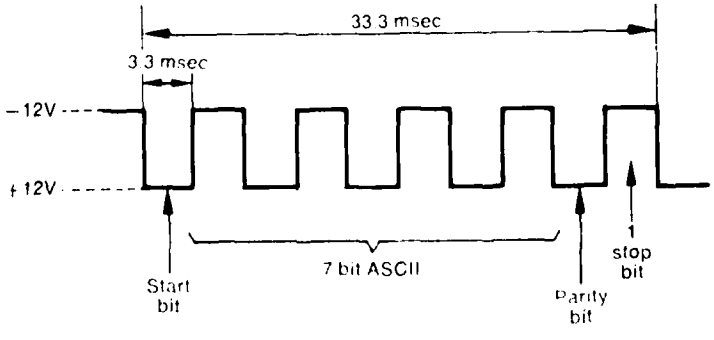


Figure 12. RS 232C format (30 characters/sec).

the program can then be run and the results examined before proceeding. The monitor program will retain all the break points until they are cleared. Also, the trace instruction, N, permits stepping through a program one instruction at a time. At the end of each trace, the registers can be examined to determine program status. Once a program is debugged, it can be written in an EPROM IC and placed into one of the preaddressed PROM sockets on the D2 PC board. That program will be executed whenever the starting address in the PROM is accessed.

PIA Interface Requirements

The Peripheral Interface Adapter (PIA) is used to interface parallel-type devices to the microprocessor. The PIA is a 40-pin IC, equivalent in complexity to the MPU itself. It provides 20 output lines to the outside world (Fig. 5); 18 of them can be either inputs or outputs. The CS lines and RS lines are connected to the MC 6800's address bus (Fig. 8), enabling the PIA to be considered as four locations in memory by the MPU. Each of the 16 peripheral data lines, PA0-PA7 and PB0-PB7, which interface with external equipment, can be programmed to act bidirectionally. The B-side output buffers have three-state capability, allowing them to enter a high-impedance state when the data line is used as an input. The eight bidirectional data lines (D0-D7) permit transfer of data to and from the MPU. The MPU receives data from the outside world from the PIA via these eight data lines, or sends data to the outside world through the PIA via the eight data lines. Interrupt control lines (CA1 and CB1) are input-only lines to the PIA, and can be programmed to set the interrupt flag of the control registers in the PIA. Peripheral control lines (CA2 and CB2) can be programmed to act either as an interrupt input or as a peripheral output for handshaking operations. As an output, these lines are compatible with standard TTL. As inputs, CA1 represents one standard TTL load, and CB1 has greater than 1-megohm input impedance.

The four address locations of the PIA access the A-side data direction register and the A-side control register and the B-side data direction register and B-side control register (Fig. 13). As previously mentioned, these addresses are decoded by the CS and RS lines of the PIA, which are connected to the MPU address bus. By programming "0"s in the data-direction register, each corresponding line performs as an input while "1"s in the data-direction register make corresponding lines act as outputs. The 16 lines may be intermixed between inputs and outputs by programming different combinations of "1"s and "0"s into the data-direction register.

The data-direction registers perform dual roles in the operation of the PIA. The first role is to determine input and output lines as described above. The second role is to function as the data registers to transfer information from the MPU's data bus to the outside world. Selection between the data registers and the data-direction registers (on either the A or B side) is made by programming a "1" or a "0" in the third least significant bit of each control register. A logic "0" accesses the data-direction register, while a logic "1" accesses the data register. At the beginning of any program, the input/output (I/O) configuration is programmed into the data direction register, after which the control register is programmed to select the data register for I/O operation.

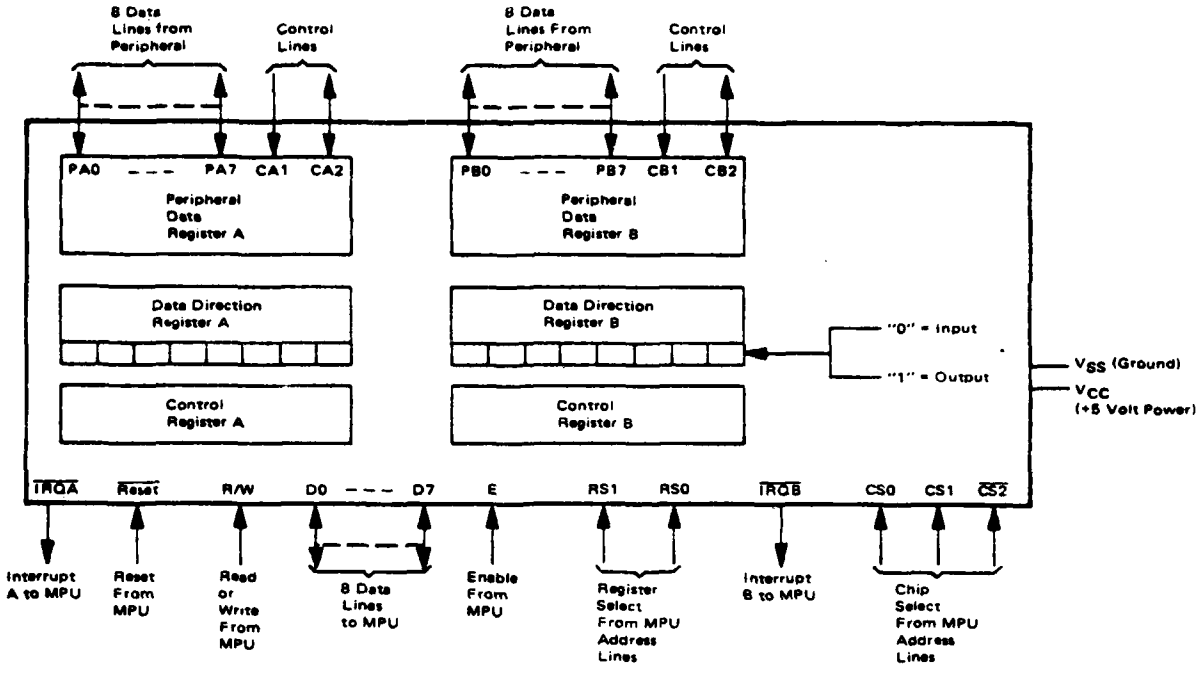


Figure 13. Peripheral Interface Adapter (PIA) functions.

An illustration of this confusing concept follows. If we design a system so that our PIA has the same address locations as the D2 evaluation module, then the following addresses are used:

8004 = A-side Data/Data-Direction Register (PDRA)
8005 = A-side control register (CRA)
8006 = B-side Data/Data-Direction Register (PDRB)
8007 = B-side control register (CRB)

Now, suppose we want to input data to the microcomputer system from a device which has eight parallel lines attached to PA0-PA7. In addition, we wish to output data from the microcomputer to another device which has eight parallel lines attached to PB0-PB7. To accomplish this, we must use a number of program instructions to configure the PIA properly.

The first step is to set the third least significant bit of the control register to a logic "0" for both A and B sides (remember, all of these internal PIA registers are 8 bits wide). Programming is as follows:

1. load the MPU accumulator A with the following 8-bits; 00000000 (clear all 8-bits of the control register),

2. send the above 8-bits to location 8005, and

3. also send the same bit pattern to location 8007.

At this point, with the control bit 2 set to a "0", we can access the data-direction registers on each side.

Now we wish to make the A-side all inputs and the B-side all outputs. To do this, we must fill the data-direction registers with all "0"s and all "1"s, respectively, e.g.,

4. load the MPU accumulator A (LDA A) with 00000000,

5. send the above pattern to location 8004 (STA A 8004),

6. load MPU accumulator B (LDA B) with 11111111, and

7. send this pattern to location 8006 (STA B 8006).

At this point we have designated PA0-PA7 as inputs and PB0-PB7 as outputs. Finally, we wish to configure our PIA so that data can be transferred to and from the MPU and the peripheral devices. We must, therefore, access the data register, not the data-direction register. To do this, we must now set bit 2 of the control registers, as follows:

8. load Accumulator A with 00000100, and

9. send the pattern to 8005 and 8006.

Now, when we refer to locations 8004 and 8006, we access the data registers of the PIA, and therefore, can transfer data to and from the data [illegible].

The above instructions are the type used to initiate and configure a system for I/O operation.

Signal Generator Interface

This section will describe the procedure necessary to interface a Wavetek Model 3001 signal generator (Fig. 14) to a Motorola D2 microcomputer evaluation module. The MPU will be programmed to scan the generator between 30 MHz and 55 MHz, and the output displayed on a spectrum analyzer as well as the digital display of the D2 module.

A block diagram of the signal generator is shown in Fig. 15. The output frequency of the Model 3001 can be externally programmed via a rear panel input connector, using standard 8-4-2-1 Binary Coded Decimal (BCD) contact closures (Fig. 16). The rear panel frequency connections are in parallel with the front panel level-indicator thumb-wheel switches. Thus, if rear panel programming is used, the front panel switches must indicate all zeros. Inputs to the pins on the rear panel connector are required to be active low by the counter's internal circuitry.

The hardware interface between the evaluation module and the signal generator is relatively simple. Wavetek supplies a rear connector with each model 3001. The PA0-PA7 and PB0-PB7 outputs on connector J1 of the D2 board are attached to pins 5 to 20 on the rear panel of the generator (Fig. 16). This allows us to externally program frequencies in steps of 0.01 MHz. Our actual application was to continuously tune from 30 MHz to 55 MHz in steps of 0.05 MHz.

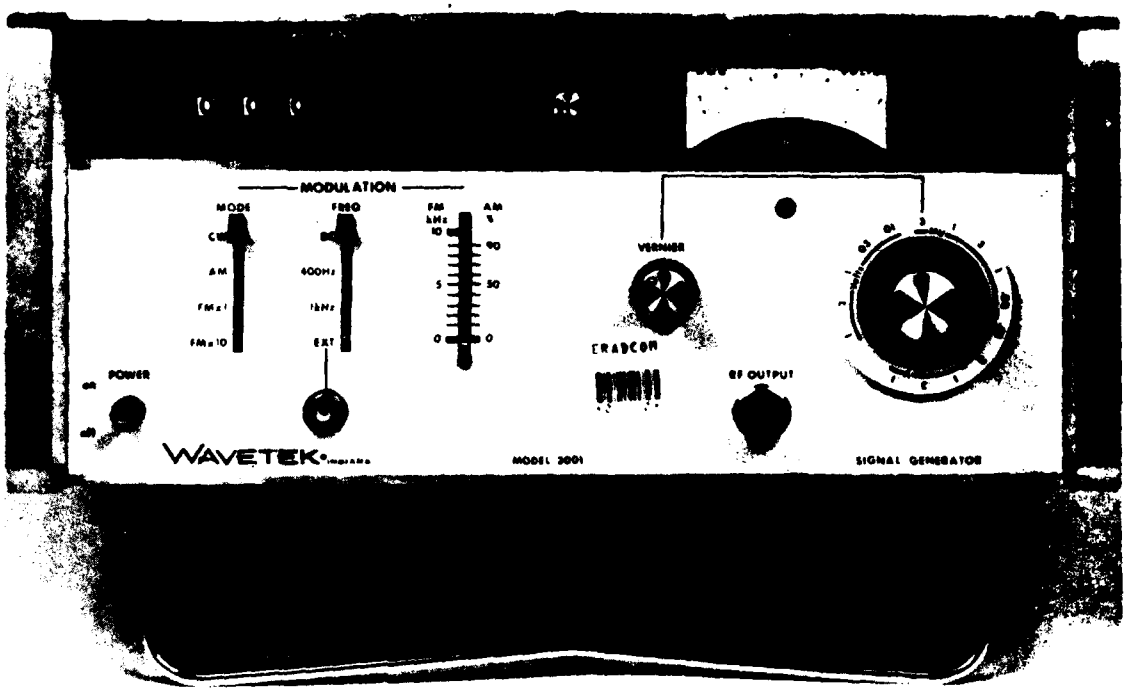


Figure 14. WAVETEK Model 3001 Signal Generator.

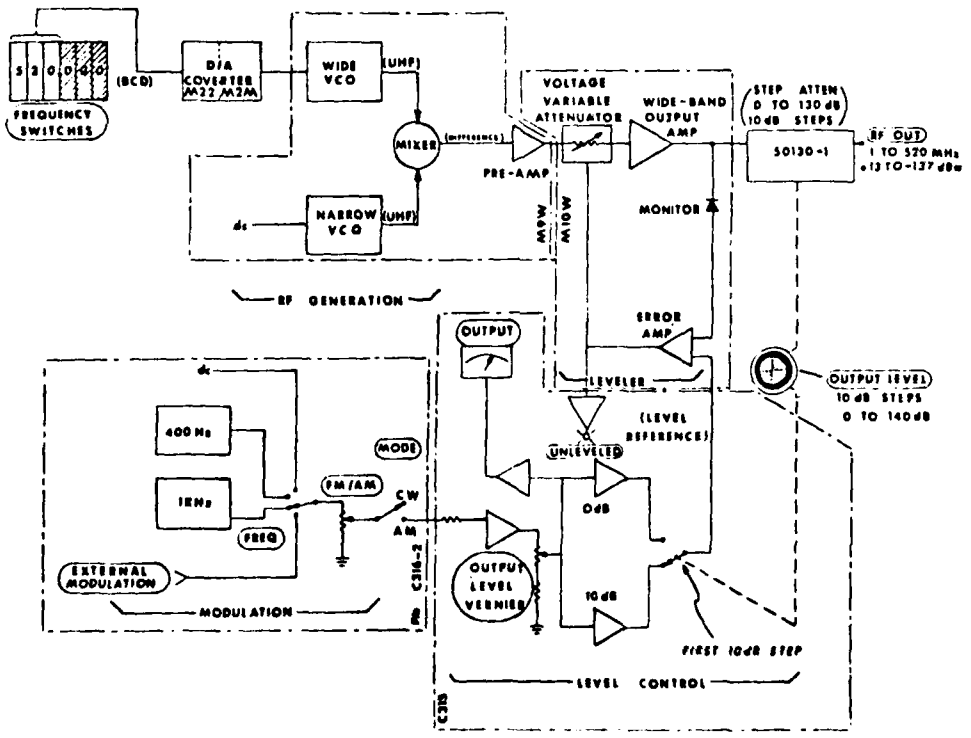
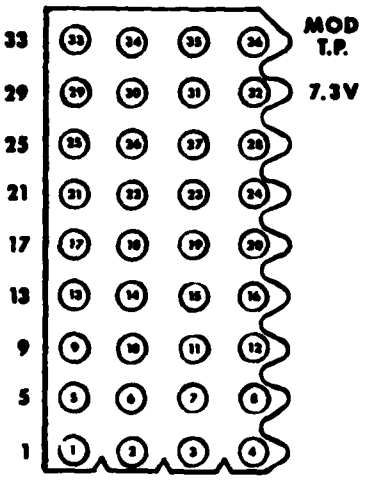


Figure 15. Signal generator block diagram.



PIN	CONNECTION	
1	N.C.	
2	400	100's
3	200	
4	100	
5	80	
6	40	10's
7	20	
8	10	
9	8	
10	4	1's
11	2	
12	1	
13	.8	
14	.4	.1's
15	.2	
16	.1	
17	.08	
18	.04	.01's
19	.02	
20	.01	
21	.008	
22	.004	.001's
23	.002	
24	.001	
25	ground	
26 - 31	N.C.	
32	7.3V	
33 - 35	N.C.	
36	MOD. T.P.	

Figure 16. Rear panel connector and pin ID.

In the software, the first step is to configure the PIA for all outputs on the A side and the B side. Data moves from the MPU to the generator at all times, and all 16 peripheral data lines are used in the interface to the Wavetek. None of the control lines of the PIA need to be connected to the generator. The MPU formats a four-digit BCD number, outputs this number to the signal generator, and delays a fixed amount of time to allow the generator's synthesizer to tune to the proper frequency value. The MPU then internally adds five to the BCD value, checks to see if it has exceeded the upper limit (55.00), and outputs the BCD number to the generator. Once the upper limit is reached, the count is reset to 30.00. The procedure has been verbally simplified, and a number of routines must be accomplished in software before completing the task. Since the MPU operates

in binary, a binary to BCD conversion subroutine was written, and a special bit rotation subroutine was put together to shift the digits into the proper places before being output to the generator.

The results of this program can be seen in Fig. 17, which shows the output signal of the generator displayed in a spectrum analyzer. The photographs are "stop action" outputs at 30, 40, 50 and 55 MHz. Actually, the MPU made the signal generator tune continuously across the band. This technique is very useful in implementing calibration procedures for sophisticated radio frequency (RF) intercept systems.

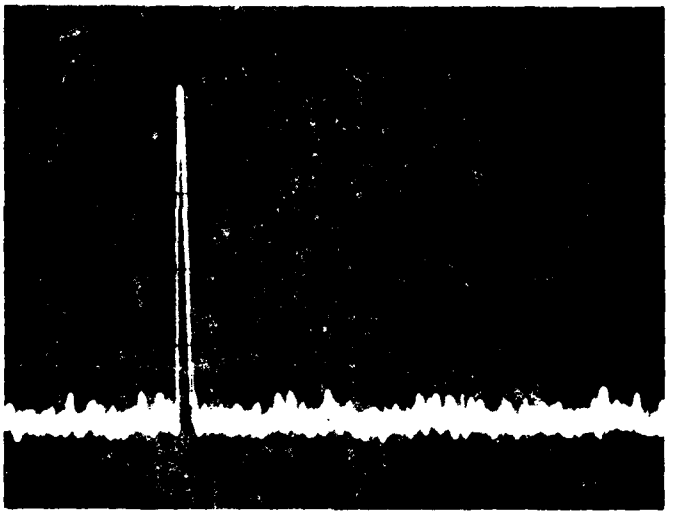
The next application is a follow-on to the original task of tuning the signal generator. A requirement existed to display the output digitally at the same time. Rather than interfacing a digital counter to the system, it was decided to simultaneously display the results on the 7-segment digits of the evaluation module display board. The software was modified to include a display subroutine, which utilizes a number of parameters contained in the monitor ROM. Some difficulties arose since the segments of each of the display digits are connected in parallel to the same lines of the display PIA. As a result, all of the digits had to be "refreshed" equally, otherwise only the most significant digit (MSD) would be visible. The problem was overcome by incorporating a 50-millisecond delay in a loop which constantly cycled through all the digits. Only software changes were needed to obtain this capability since the evaluation module is designed with the display board already interfaced to the microcomputer through a separate connector and PIA (Fig. 11).

Printer Interface

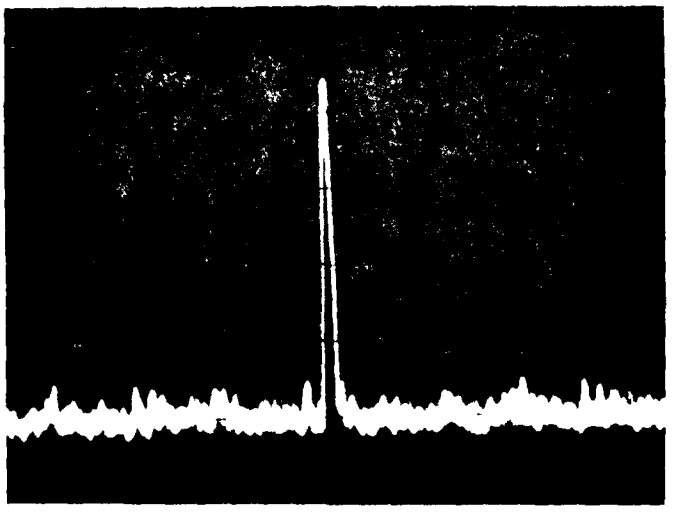
Many of today's processing applications require some sort of "hard copy" capability for recording results. This is especially true for those tasks in which remote or unattended testing takes place. Often it is not practical to have a terminal available, and a relatively inexpensive printer is adequate to perform the hard copy function. This section will describe the methods involved in interfacing digital panel printers to the MEK 6800 D2 evaluation module.

For numerical applications, the Datel DPP-7 digital panel printer (Fig. 18) is a suitable piece of hardware. A block diagram of the printer is shown in Fig. 19. The DPP-7 is a seven-column, panel-mounted terminal which can output three lines of data per second. The output data can either use a leading sign and six-decimal-digits format, or else two leading identifiers and four data digits. The printer accepts 24 input lines, TTL compatible, which can be positive or negative true-level sensitive. The inputs must be full parallel BCD (1-2-4-8) in order to result in six-decimal-digit outputs.

The usual method of interfacing the DPP-7 to a microcomputer is to use handshaking between the printer and the MPU. This means that the printer will inform the MPU when it is ready to accept data, and the MPU will inform the printer when it has sent the data. Therefore, coordination is total and

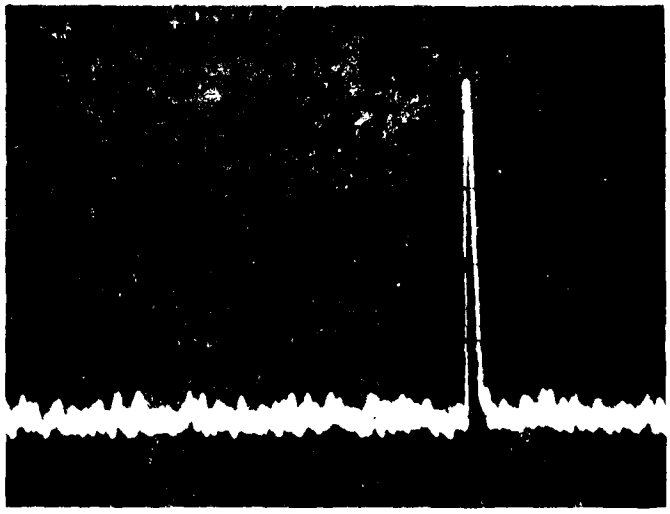


(a) 30 MHz

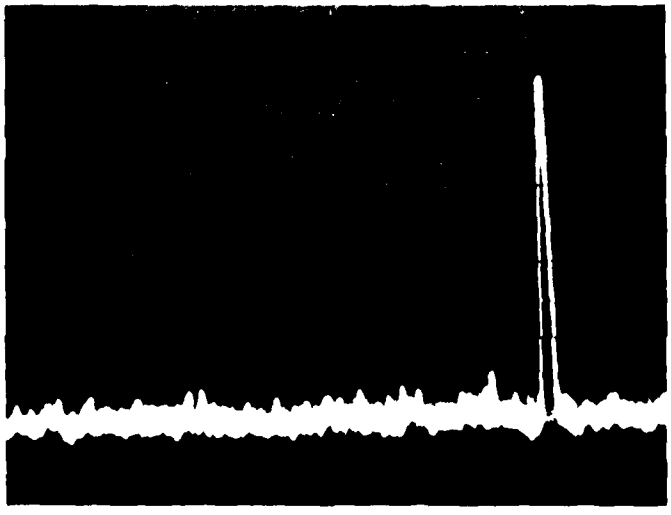


(b) 40 MHz

Figure 17. Signal generator outputs.



(c) 50 MHz



(d) 55 MHz

Figure 17. Signal generator outputs.

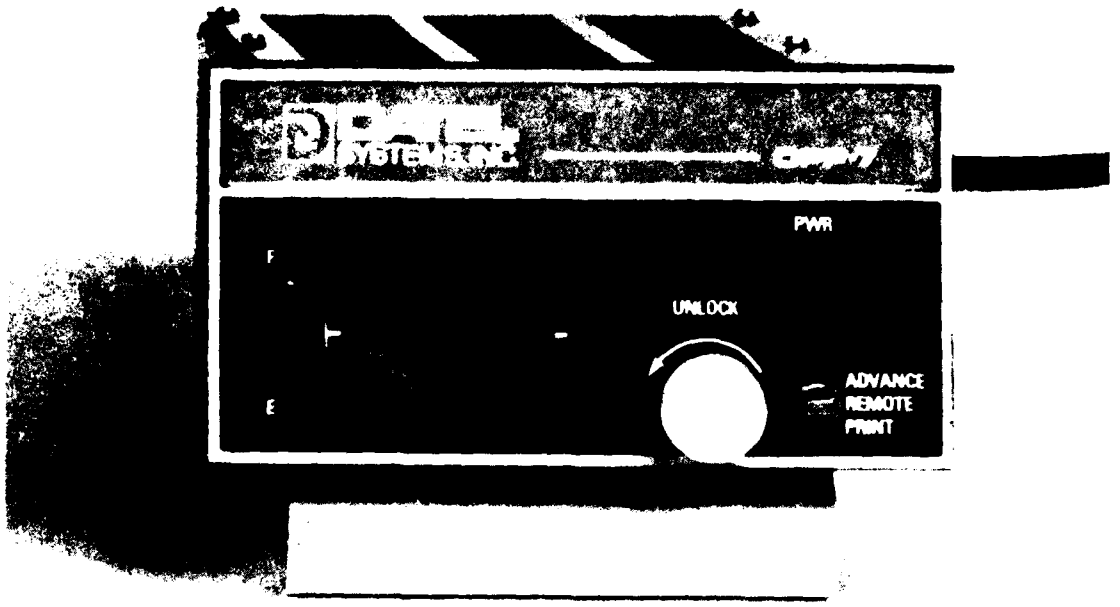


Figure 18. DPP-7 Digital Panel Printer

complete, and no data will be lost in the process. The control lines of the PIA are used to implement handshaking operations.

The Print and Busy lines of the DPP-7 (Fig. 19) are used to handshake with the microcomputer. The Busy signal is an output from the printer that can be positive or negative TRUE, and it remains TRUE during the print cycle. The data inputs may be changed 500 nanoseconds after transition to TRUE. The next Print command can be enabled when the Busy line goes FALSE. The Print signal from the microcomputer must be a pulse that is from 1 microsecond to 200 milliseconds wide. The data to be transferred must be valid 500 nanoseconds before the Print pulse and 1 microsecond after its leading edge.

The actual handshake operation will commence with the DPP-7 sending a high-to-low transition on the Busy line to CB1. The PIA informs the MPU that the printer is ready for data by generating an interrupt on IRQB (see Fig. 13). The MPU sends the data out on the data bus and tells the printer to print, using a pulse of the proper width on CB2 of the PIA. The CB2 is connected to the Print line of the DPP-7. The printer accepts the data and pulls the Busy line high while it is printing. Sometime after system initialization, the PIA must be configured properly so that the above scenario can occur.

As an example, let us assume that the B side of the PIA has already been designated as all outputs. We must now address the B-side control register so that handshaking can take place. We accomplish this as follows:

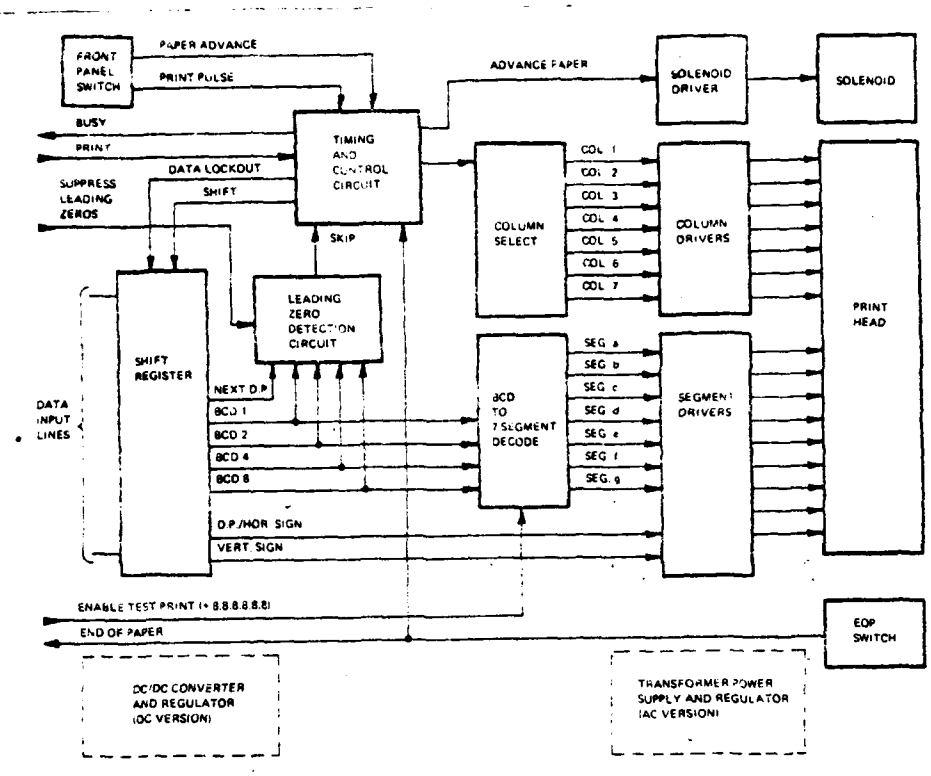


Figure 19. DPP-7 block diagram.

(a) load Accumulator A with 00100101, and (b) send this bit pattern to location 8007. Recall that 8007 is the location of the B-side control register (CRB) of the PIA. Figure 20 shows that a "1" in bit 2 of CRB will access the Data register on the B-side so that information can be transferred. A "1" in bit 5 of CRB designates the CB2 line as an output, and a "1" in bit 0 of CRB allows an interrupt to occur when CB1 goes low. Our system is now prepared to transfer data in the handshaking mode.

When the printer's Busy line goes low, indicating that it is ready for data, the CB2 line will go high; it will then go low again after the MPU sends data to the B side of the PIA. This is effectively a pulse on the DPP-7's Print line, which forces the Busy line high after the printer accepts the data. The MPU will be formatting its next set of data and will wait for a subsequent interrupt from the DPP-7.

There are many situations which arise when a design engineer needs more than numerical information as an output. For these applications, an alphanumeric printer, such as the Datel APP-20 panel printer, can be used. The APP-20 prints the full ASCII (American Standard Code for Information Interchange) character set of upper and lower case letters, numerals, punctuations, etc., in 20 columns on thermal paper. For users who wish to upgrade to the fully alphanumeric unit, the APP-20 uses the identical panel mounting and outline dimensions as the DPP-7. A block diagram of the APP-20 is shown in Fig. 21.

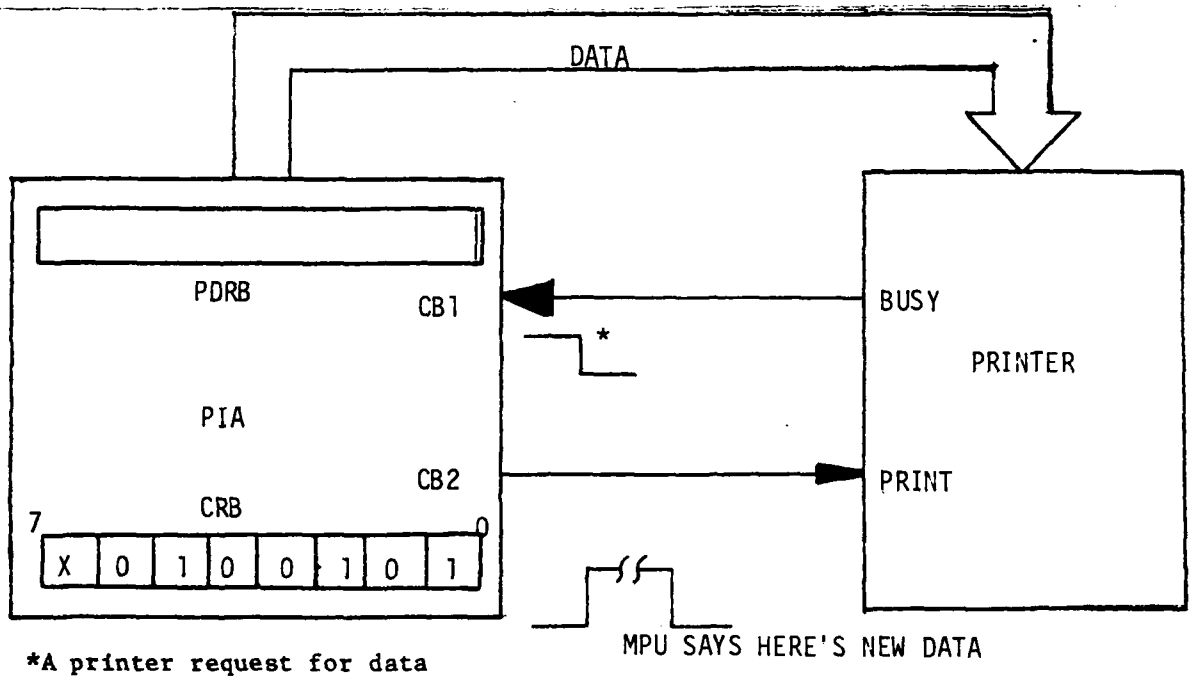


Figure 20. Handshaking with the printer on the "B" side.

This printer is more sophisticated than the numerical printer and is more difficult to interface to a microcomputer. The difficulties arise in software control, however, not in hardware implementation. The input/output connection to the APP-20 consists of data and control lines configured on a single 25-pin type "D" rear connector. The pin connections are listed in Table 1. Seven data inputs select upper and lower case ASCII characters, and the eighth bit designates programmable formatting characters. Additional control lines form an interlocked asynchronous handshake to enter each character into a 20-column input register (Table 2). Characters are loaded sequentially as 8 bits in parallel, and printing begins after all desired characters are loaded into the register. The printing rate is approximately 1-1/2 lines per second.

Table 1. APP-20 I/O Pin Connection

PIN	FUNCTION	PIN	FUNCTION
14	Text/Lister Norm/InV-Input)	1	Ready for Data (Output)
15	Single Char Print (Input)	2	Data Accepted (Output)
16	Tall Characters (Input)	3	Input Register Full (Output)
17	Data Valid (Input)	4	Not Used
18	Data Bit 2 (Input)	5	+5 Vdc Power Out, .5A max.
19	Data Bit 3 (Input)	6	Not Used
20	Data Bit 1 (Input)	7	Logic Ground
21	Data Bit 0 - LSB (Input)	8	Not Used
22	Data Bit 4 (Input)	9	Not Used
23	Control Characters - MSB (Input)	10	Not Used
24	Data Bit 5 (Input)	11	Print (Input)
25	Data Bit 6 (Input)	12	Data POS/NEG True (Input)
		13	End of Paper (Output)

Table 2. APP-20 I/O Interface Connections

Function	Name	No. of Lines
Data	ASCII characters	7 lines
	Control Bit	1 line
Mode Inputs (if used, internal pull-ups supplied)	Data Pos/Neg. True	1 line
	Text/Lister	1 line
	Single Char. Print	1 line
	Tall Characters	1 line
Handshake Inputs	Data Valid	1 line
	Print	1 line
Handshake Outputs	Ready for Data	1 line
	Register Full	1 line
	Data Accepted	1 line
	End of Paper (Sw. to gnd.)	1 line
Logic Ground		1 line

The timing and control requirements of the APP-20 are very strict and precise. A timing diagram for printing one line is shown in Fig. 22. The hardware interface between the evaluation module and the printer is on a cable from the 25-pin connector of the APP-20 to the J1 connector of the D2 board. Handshaking takes place with the CB1 line of the PIA connected to DATA ACCEPTED and the CB2 line attached to DATA VALID. The REGISTER FULL output from the printer is connected to its own PRINT input, so that printing automatically takes place after 20 characters are input. However, the number of characters per line must be counted by the MPU. Once the 20 character buffer is full, the REGISTER FULL output pulse will signal the PRINT input, and the actual printing will take place. During this period, the DATA VALID line will be high (Fig. 22), so that no additional pulses will interrupt the MPU on CB1.

An APP-20 initialization routine is necessary for power-on reset operation. The MPU informs the printer that the first character is ready to be loaded by using a DATA VALID falling edge input on CB2, which will load the data. This is done by configuring the PIA for pulse output operation. The APP-20 will respond within 300 microseconds with a DATA ACCEPTED output, telling the MPU to update the bus with data for the second character. Prior to this 300-microsecond response, the PIA is reconfigured for handshake operation for the rest of the printing cycle.

Figure 23 describes the method for configuring the PIA for outputting a pulse to the printer from CB2. A "1" in bit 5 of the CRB designates CB2 as an output; a "1" in bit 3 of the CRB means pulse mode; and a "1" in bit 2 of the CRB accesses the data register for information transfer. The output pulse will occur whenever the MPU sends data out to the B side of the PIA. As an example, let 8006 = B-side Data Register address, and 8007 = B-side Control Register Address. The procedure would then be as follows:

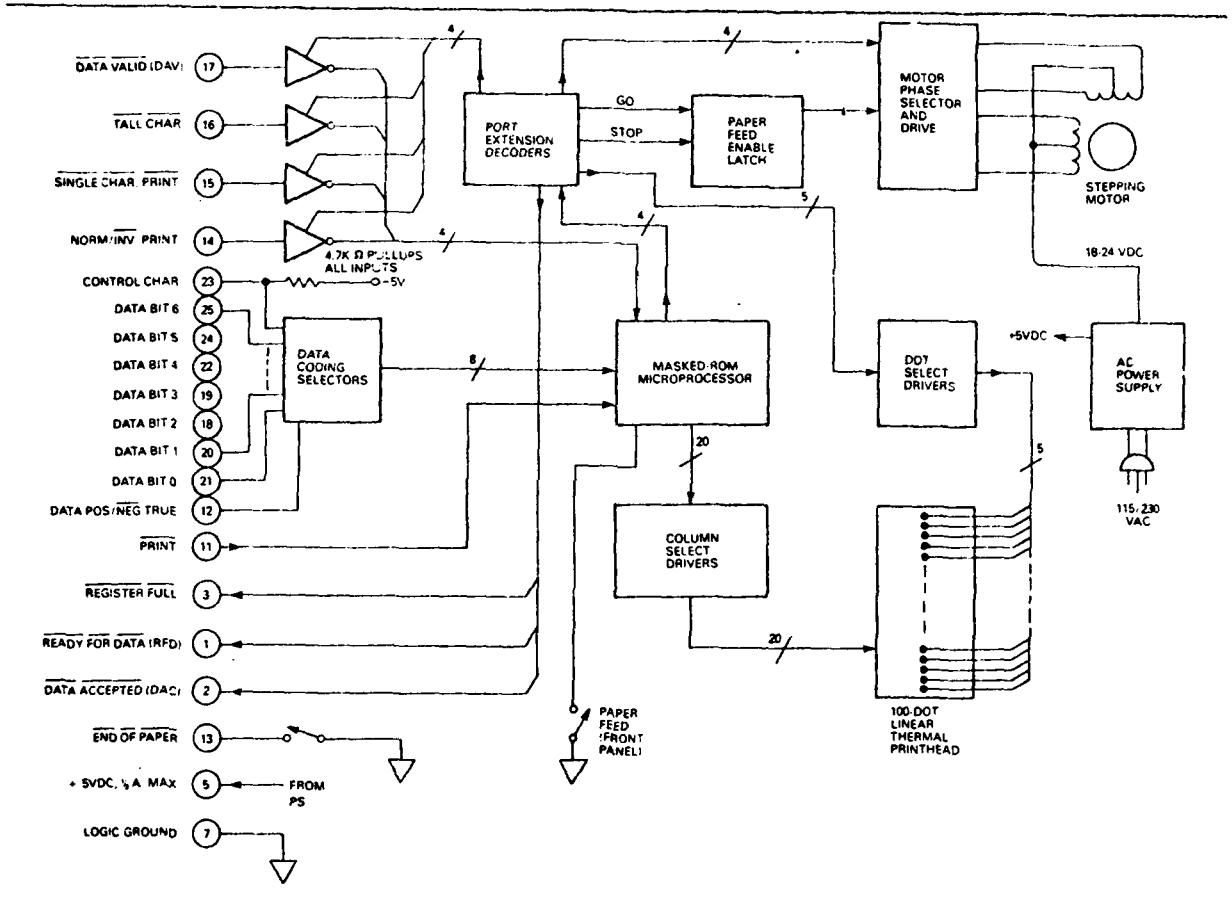


Figure 21. APP-20 block diagram.

1. load Accumulator A (LDAA) with 00101100 (2C - the Hexadecimal representation of the binary number), and

2. send the bit pattern in A (STAA) to 8007.

Now the PIA is configured to send a pulse out on the CB2 line whenever instructed to do so by the MPU. Suppose the data to be printed were all "1"s; then,

3. LDAA 1111 1111 (FF), and

4. STAA 8006.

As soon as the MPU executes the above statement, the CB2 line, which will normally be high, goes low on the positive transition of the MPU enable pulse. Then CB2 will go high again on the next positive MPU enable pulse transition. The net effect is a CB2 pulse, as shown in Fig. 24. The width of the minimum CB2 pulse depends on the width of the enable pulse, which in turn depends on the system clock rate. The clock of the D2 evaluation module operates at a 614.4 kHz rate, so the CB2 minimum pulse width is just under 2 microseconds. The data in the data register of the PIA is latched and available on the first negative transition of the CB2 pulse. The repetition interval between pulses is totally under control of the MPU. One can vary the intervals in software by using different instructions to change cycle times. This is illustrated in Fig. 25.

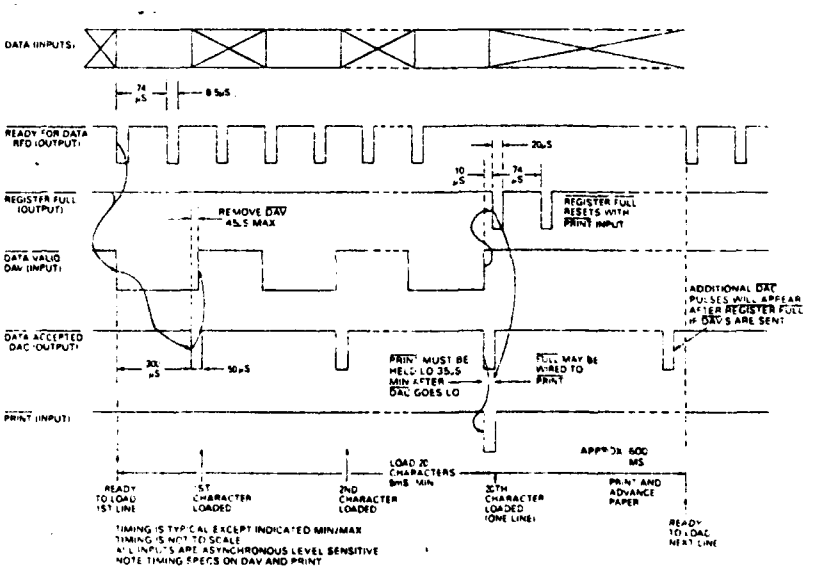


Figure 22. APP-20 timing, printing one line.

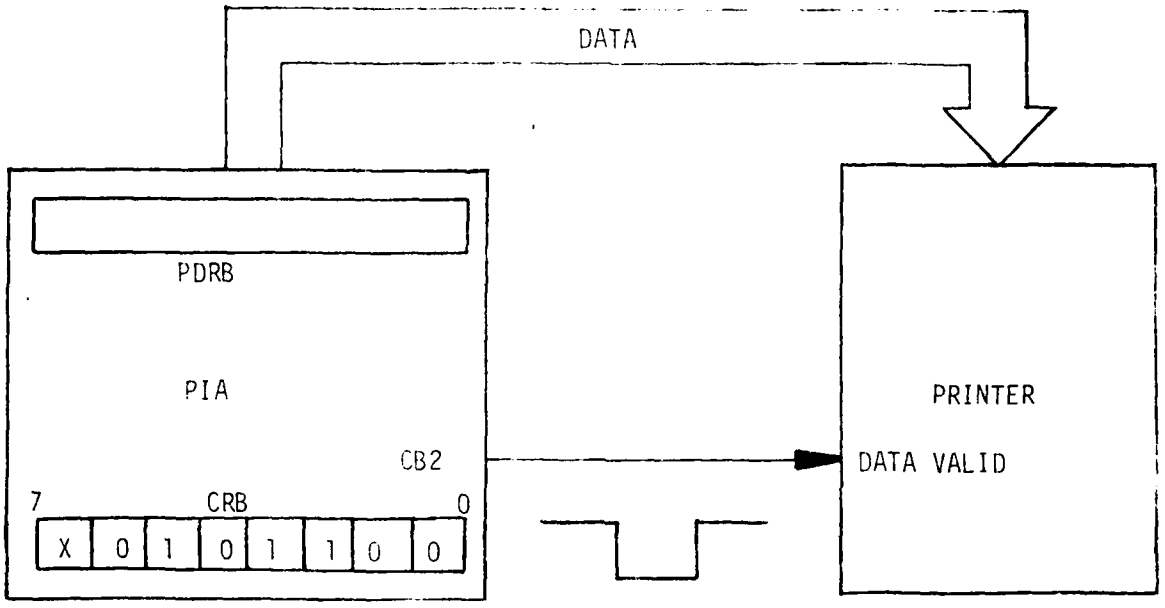
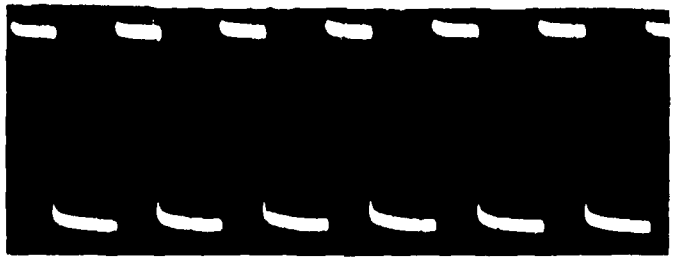
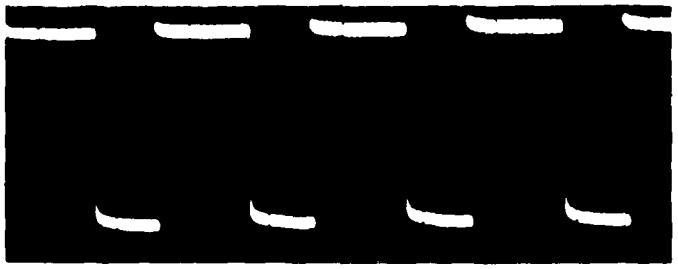


Figure 23. Pulse output on "B" side configuration.



(a)



(b)

Figure 24. CB2 outputs in pulse mode.

The APP-20 printer uses its own internal microprocessor to control data transfer and printing operations. During the handshake mode, once the printer has accepted an ASCII character and the DATA ACCEPTED line goes low, our system must bring the DATA VALID line high within 45 microseconds. If not, the APP-20 will poll the DATA VALID line again and load whatever is on the data bus as the next character, whether the bus is ready or not. This 45-microsecond requirement is easily taken care of in software, but the programmer must always be aware of it.

Data Acquisition Systems

The previous section has described typical printer interfaces for microcomputer applications. In general purpose computing tasks, the MPU may perform sophisticated, mathematical operations and dislay the results, obviating the need for any other input or output interfaces. More often than not, though, the system is connected to the outside world and must analyze a particular measurement. This is the classic dedicated control application of an MPU-based system. The microcomputer is the controller of the experiment and the analyzer of the data it obtains.

Most instruments and sensors of the real world are analog in nature. Microcomputers operate using digitally formatted data. The interface to the real world, therefore, will involve the analog-to-digital (A/D) conversion of information.

Although there are various methods of A/D conversion, each system can usually be divided into two sections; an analog subsystem containing the various analog functions for the A/D, and a digital subsystem containing the digital functions. To add an A/D to the MPU, both of the sections may be added externally to the microprocessor in the form of a PC card, hybrid module, or a monolithic chip.

Two A/D conversion architectures that can be found in 90% of all converters sold are successive approximation (S/A) and dual-slope integration. The S/A technique uses a digital-to-analog converter (DAC) in a feedback loop to generate a known analog signal to which the unknown analog input is compared (Fig. 26). The known signal out of the DAC is actually a group of weighted binary references, equal to the number of bits in the output code. The successive approximation method is usually used in high-speed applications.

The second A/D conversion method, dual slope, is an integration method in which the conversion is accomplished by first charging an integrator, and then discharging it back to zero. This technique is also called the dual-ramp technique. The ratio in time-of-the-ramp lengths provides a value representing the difference between a reference and an unknown voltage.

(c)

(b)

(c)

(d)

Figure 25. CB2 inverted pulses at various levels.

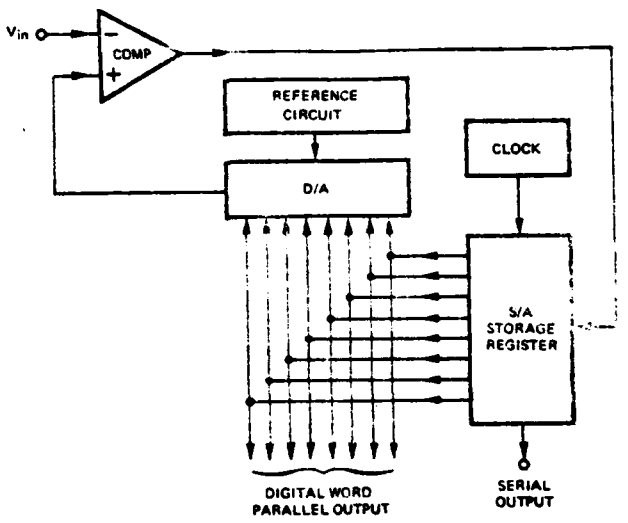


Figure 26. Successive Approximation A/D Converter

The basic waveforms for the dual ramp A/D are shown in Fig. 27. During time period T1, the unknown input is integrated for a fixed time period (fixed number of clock cycles). The integrator voltage increases from the reference level to a voltage which is proportional to the input voltage. At the end of this time period, a reference voltage is applied to the input of the integrator, causing the integrator output voltage to decrease until the reference level is again reached. The number of clock cycles that are required to bring the integrator output voltage back to the reference level (T2) is proportional to the input unknown voltage. This approach has a longer conversion time than that of the successive approximation method. However, this method usually performs a more accurate A/D conversion, and at a much lower cost.

The A/D conversion is the heart of a data acquisition system, but it is by no means the only necessary component. Most acquisition systems contain one or a number of sample-and-hold (S/H) registers to latch analog information from a particular source. Also, many systems contain a multiplexer (MUX), which allows a single interface to access many sources of data. The timing and control of all of this hardware can then be provided by a microcomputer.

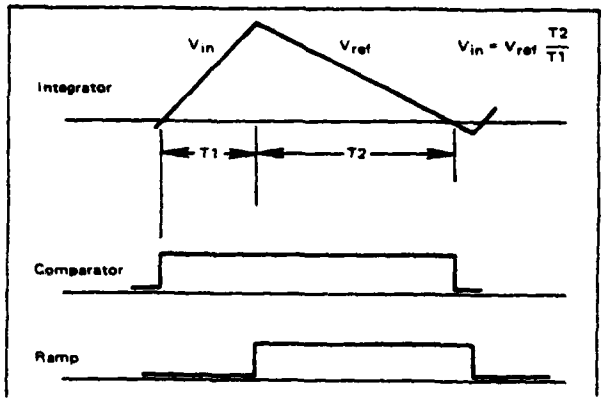


Figure 27. Dual-slope integrating waveform.

The data acquisition system discussed in this report consists of multiple sample and hold circuits feeding a MUX, which in turn transfers the information to an A/D converter. Real-world situations require that some sort of signal conditioning is necessary between the S/H and the sensor or transducer. Usually filters and isolation amplifiers are needed to insure the proper transfer of data. This aspect of the acquisition process will not be covered at this time. A key component in our data acquisition system is the S/H circuit, which is designed to be digitally commanded to operate in either of two modes: the SAMPLE mode, or the HOLD mode.

In the SAMPLE mode, the S/H circuit output reproduces the input very closely; in fact, ideal SAMPLE-mode behavior is characterized by output tracking and reproducing the input perfectly. Figure 28 is the block diagram of a typical S/H circuit.[2]

In the HOLD mode, the output maintains very closely the value it had when the S/H received the command on Pin 9 to transfer to the HOLD mode. In practice, the amplifier loads a capacitor, whose output must decay, however slowly. When the capacitance is increased, the decay rate or droop is slowed. However, the acquisition time for a given switch impedance is increased. The S/H droop rate actually determines how infrequently a channel may be sampled in a system and, therefore, establishes the minimum throughout. The S/H must hold a sampled signal within ± 1 LSB (least significant bit) for as long a time as it takes the A/D converter to convert all other inputs sampled at the same time. A very low initial droop rate is important, since droop is temperature sensitive and typically doubles every 10^{o} C.

2. Datel/Intersil Engineering Product Handbook 1977, Vol. 3, p 122, (1976-1977).

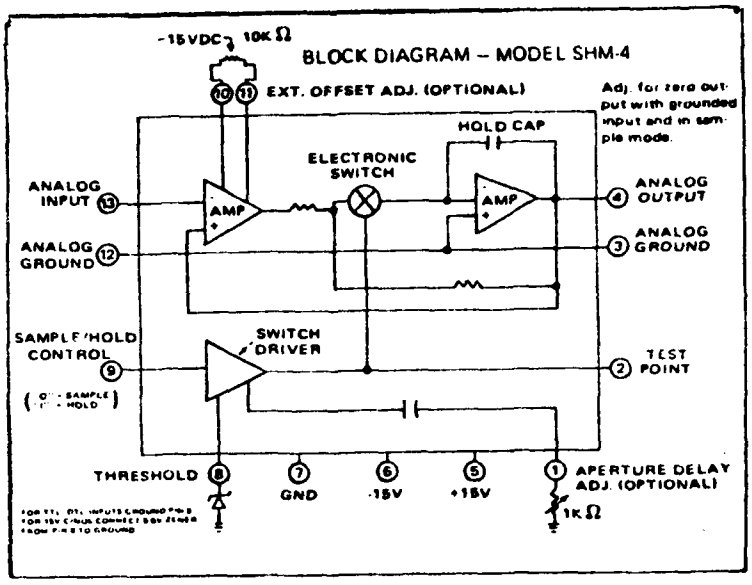


Figure 28. SHM-4 sample and hold circuit.

The S/H gain, linearity, noise, and harmonic distortion performance affect the system accuracy during the SAMPLE mode. The above parameters determine the I/O transfer characteristics while in the tracking phase. The S/H slew rate and settling time affect the system throughput capability, thus determining a minimum time to be allocated to the acquisition phase. During the S/H transition, the signal being held for conversion may differ from the signal at the instant of the HOLD command by some error. This is due to the fact that the S/H requires a finite time in which to open the switch and to disconnect the input buffer from the holding capacitor. The error depends upon the dynamics of the signal. During the hold-to-sample transition, the acquisition time is defined as the amount of time required, from the instant the sample command is given, for the output to first reach and settle within a stated tolerance. The acquisition time of the S/H affects the overall system throughput rate and is a measure of the minimum delay required before conversion can take place for a given input signal.

A MUX is a circuit designed to connect any number of input signal paths to an output load, switching from path to path in an arbitrary, consecutive, or changing sequence. This is accomplished in accordance with a digitally coded "address" instruction from the system program control source (in our case, the microcomputer). Most MUX modules contain internal decoding circuitry to convert the binary address input to one-in-four, one-in-eight, or whatever else is required to switch on one channel at a time. Figure 29 shows a typical N-channel MUX with a single-ended output. Among the key parameters for MUX selection are: the number of channels, capability for expansion, voltage rate for linear operation, and impedance and configuration (single-ended, balanced differential, etc.). Dynamic considerations include small-signal bandwidth, slew rate, settling time, full-power bandwidth, and recovery time.

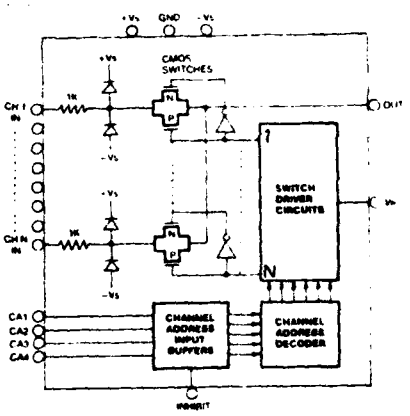


Figure 29. N-Channel multiplexer circuit.

The A/D converter used in our system is the ADC-149 from Datel[3] (Fig. 30). It is a 14-bit successive approximation converter designed mainly for original equipment manufacturer (OEM) uses. It was specifically made to provide high resolution and accuracy for incorporation into precision instruments for process control systems and test and measurement systems. The converter accepts either unipolar or bipolar input voltages and performs a 14-bit conversion in 50 microseconds. Several output codes are available, including straight binary for unipolar inputs, and either offset binary or two's complement for bipolar inputs.

We mentioned in the previous section that the "B" side of a PIA is normally used to transfer data from a 6800-based system because of the drive capabilities of the "B" side data and control output lines. We followed this method for the printer interfaces. Conversely, the "A" side data and control lines of the PIA are normally used to <u>input</u> data to the system from the outside world. This would be fine for our data acquisition system, if we had an 8-bit A/D converter. Since the ADC-149 is a 14-bit converter, we shall input data both on the "A" and "B" sides simultaneously.

Let us assume that our data input system consists of four channels of information feeding four S/H circuits, which in turn are connected to a 4-to-1 MUX. The MUX outputs a single channel of information to the ADC-149, which does a conversion in 50 microseconds. Let us now initialize our

3. Datel/Intersil Design Engineers Handbook Goldbook Vol. 3, Hayden Publishing Co., Inc., (1980).

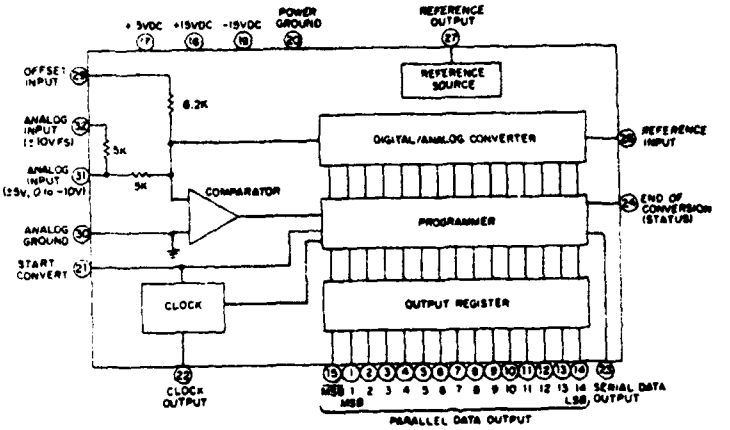


Figure 30. ADC-149 S/A A/D Converter.

microcomputer system to handle the data flow and then to perform the timing and control functions.

Since this particular application requires the movement of more than eight bits of information at one time, we implement a change in the addressing scheme of the particular PIA in use. We normally tie the RS0 line to A0 of the address bus, and RS1 to A1 (Fig. 13). For this case, we reverse the connections so that RS0 is connected to A1 and RS1 to A0. This will place the peripheral data registers and control registers side by side in the memory map as follows:

8004 = A-side Data Register,

8005 = B-side Data Register,

8006 = A-side Control Register (CRA) and

8007 = B-side Control Register (CRB).

The 16-bit index register (X) can now be used to transfer data with one instruction, instead of the 8-bit accumulator (A). We shall use a second PIA, conventionally addressed, to maintain timing and control of the S/H circuits and the MUX. This PIA occupies addresses 8020-8024. Let us initiate our system by clearing bit 2 of the PIA control registers to access the data direction registers. Recall that "1"s will designate outputs, and "0" inputs; then

LDAA 00,

STAA 8006 Clear CRA (PIA1),

STAA 8007 Clear CRB (PIA1),

STAA 8021 Clear CRA (PIA2) and,

STAA 8023 Clear CRB (PIA2).

Now that we have accessed the data direction registers, let us designate the inputs and outputs. Thus,

LDX 0000 (all "0"s i.e., all inputs),

STX 8004 (loads A&B direction registers for A/D),

LDAA FF (all "1"s i.e., outputs), and

STAA 8022 (loads B direction register).

The last statement designates the B side of PIA2 as all outputs. We will use these lines to control the S/H circuits and the MUX.

We have finished determining the direction of our flow of information, so we now set bit 2 of the control registers to access the Data Register. Therefore,

LDAA 04 (i.e., 0000 0100 - set Bit 2),

STAA 8006 (Bit 2 set for CRA of PIA1),

STAA 8007 (Bit 2 set for CRB of PlA1), and

STAA 8023 (Bit 2 set for CRB of PIA2).

Like many other peripheral circuits, the ADC-149 converter should exchange data with the MPU in a handshaking mode. For this case, the A/D converter should signal that it has accepted the data. Figure 31 shows the correct PIA configuration for transferring data from a peripheral to the MPU, using handshaking. The CA2 line of the PIA is connected to the A/D converter's "start of conversion" line, and the CA1 line is connected to the "end of conversion" line, so

LDAA 25 (i.e., 0010 0101) and,

STAA 8006 ("A" side handshake),

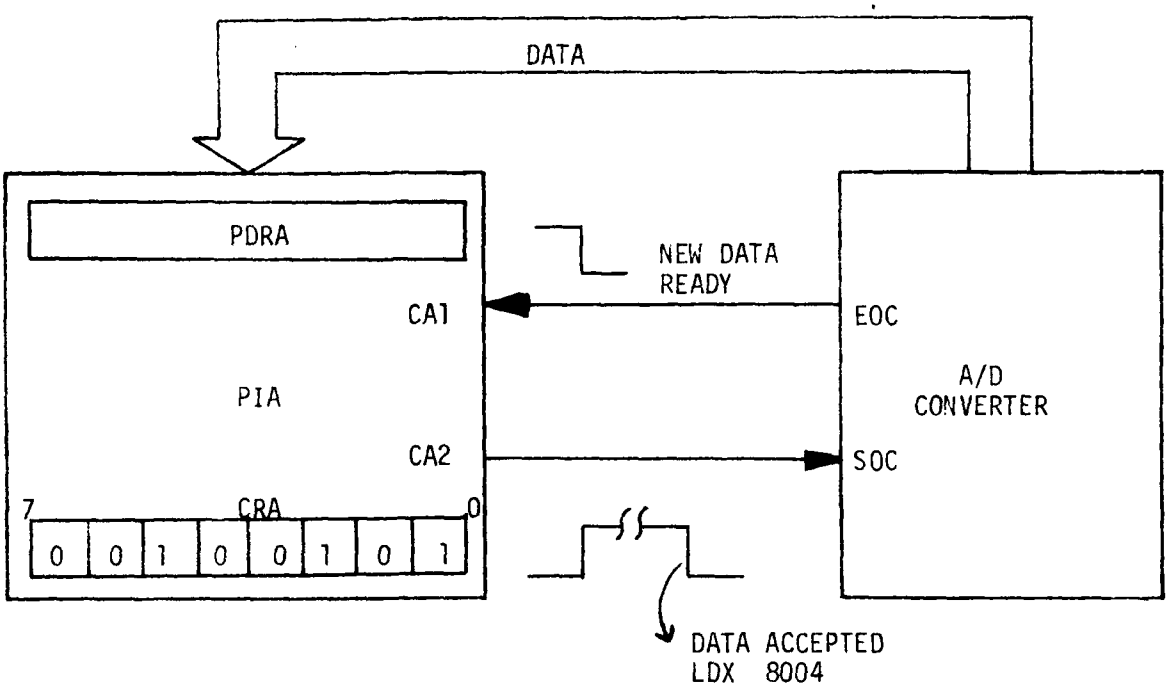


Figure 31. Handshaking with peripheral on "B" side.

When the ADC-149 has completed a conversion it will send out a high-to-low transition to CA1, which will cause an interrupt of the MPU. The MPU will then output a low-to-high transition on CA2, telling the ADC-149 to start the next conversion. The CA2 line will go low again after the data on the "A" side has been read by the MPU, i.e.,

LDX 8004 (Read 16 bits; data accepted).

We are now able to input digital data from the converter, but we must also determine the correct procedure for doing so. Let us designate the four S/H circuits as SH1 to SH4. From Fig. 28 we see that a zero on pin 9 means sample, and a one means hold. If each of the control lines is tied to A-side data lines PA0-PA3 (Fig. 15) of PIA2, we can easily control the S/H circuits by sending zeros or ones out from the MPU. For example, if we wish SH1 to SH4 to all sample simultaneously, then

LDAA XXXX0000 all zeros (X-represents a "don't care" condition), and

STAA 8020 (all 4 sample mode).

Also, if we wanted SH1 to sample and the rest to hold, then

LDAA XXXX1110, and

STAA 8020 (SH1 samples, rest hold).

In a similar manner, we can control the channel selection of the MUX by inputting the proper values of ones and zeroes on CA1 and CA2 (Fig. 31). Table 3 gives the channel addressing for an MX-409 4-to-1 MUX.

TABLE 3. MUX CHANNEL ADDRESSING

Inhibit	CA2	CA1	ON CHANNEL
0	X	X	None
1	0	0	1
1	0	1	2
1	1	0	3
1	1	1	4

Suppose that CA1, CA2, and inhibit were connected to PA4, PA5, and PA6, respectively. Now let us sample all four S/H circuits simultaneously while inhibiting the MUX.

LDAA XOXX 0000 Sample all, and

STAA 8020 (inhibit MUX).

If we wish to latch the data for all channels and choose, say channel 3, to put through the MUX, then

LDAA X1101111, and

STAA 8020 (hold all, pick CH3).

In order to convert the value from channel 3, we would now send out a start of-conversion pulse to the ADC-149 and wait at least 50 microseconds for the end-of-conversion pulse, then read the digital data as previously described. It's easy to see that the microcomputer can maintain total control of the data collection process. Once the digital information is

acquired, it can be processed and then outputted to a display or printing mechanism through an interface, as described in the previous section.

CONCLUSIONS

The important factor in interfacing a Motorola 6800 microprocessor to the outside world is through a thorough understanding of the requirements of the Peripheral Interface Adapter integrated circuit. Proper addressing and control of this IC will enable a system designer to implement many different interfacing techniques.

The PIA was configured for a particular system by accessing its internal registers. The data registers, control registers, and data-direction registers were accessed by using a specific set of assembly language instructions. These instructions allowed information transfer along the system data bus to the addresses occupied by the PIA.

The signal generator, printers, and data acquisition system were all tied to the output lines of a single PIA or multiple PIA's. Hardware requirements of these external devices are the same, but the number of lines required by each is different; i.e., they each require their own type of MPU instructions for a specific PIA configuration. However, the address requirements of the PIA's themselves remain the same. The transfer of data to and from the processor is then reduced to writing and executing a control program.

BIBLIOGRAPHY

L. Leventhal, L.A., 6800 Assembly Language Programming, Osborne & Associates, Inc., Berkeley, CA, 1978.

2. Lesea A., and Zaks, R. A., Microprocessor Interface Techniques, Sybex Inc., 1978.

3. M6800 Microprocessor Applications Manual, Motorola Semiconductor Products, Inc., McGraw Hill, NY, 1975.

4. M6800 Microprocessor Programming Manual, Motorola Inc., 1975.

5. Osborne, A., An Introduction to Microcomputers, Vol 0, Osborne & Associates, Inc., Berkeley, CA., 1976.

6. Peatman, John B., Microcomputer Based Design, McGraw Hill, NY, 1977.

7. Hilburn, J., and Julich, P., Microcomputer/Microprocessors, Hardware, Software and Applications, Prentiss Hall, Englewood Cliffs, NJ, 1976.

8. Datel/Intersil Engineering Product Handbook Gold Book 1979/1980, Vol 3, Hayden Publishing Company.